(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 February 2001 (08.02.2001)

PCT

(10) International Publication Number
WO 01/09714 A2

(51) International Patent Classification[7]: G06F 9/00

(21) International Application Number: PCT/US00/19552

(22) International Filing Date: 28 July 2000 (28.07.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/364,170 29 July 1999 (29.07.1999) US
09/363,978 29 July 1999 (29.07.1999) US

(71) Applicant: RESPONDTV, INC. [US/US]; 1083 Mission Street, San Francisco, CA 94103 (US).

(72) Inventors: WEBER, Jay, C.; 302 Pope Street, Menlo Park, CA 94025 (US). LASH, Todd; 6668 Colton Boulevard, Oakland, CA 94611 (US). STEFANAC, Suzanne; 3435 Cesar Chavez, Suite PH, San Francisco, CA 94110 (US).

(74) Agent: DONNELLY, Darren, E.; McCutchen, Doyle, Brown & Enersen, LLP, Three Embarcadero Center, San Francisco, CA 94111 (US).

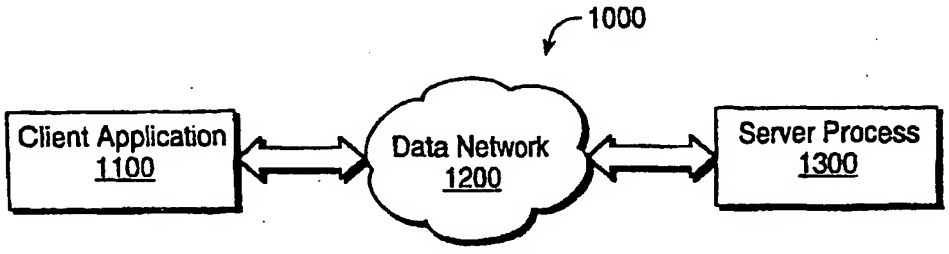
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: METHOD, APPARATUS, AND COMPUTER PROGRAM PRODUCT FOR DEFERRED COMPLETION OF MULTI-STEP USER TRANSACTION APPLICATIONS



(57) Abstract: Disclosed are server features for allowing a client user to defer completion of multi-step user transaction applications ("MUTAs") and later resume the MUTA. One disclosed feature is providing a selectable deferral action in a page implementing part of the MUTA, e.g., HTML, XML, JavaScript/ECMA Script document. If the client user selects to defer completion, a state object is created by the server and stores state information including information previously entered during the MUTA. The server provides a resumption object to the user including a resource for resuming the deferred MUTA. When the user selects the resource, state information stored in the state object is loaded and the use may complete the remainder of the MUTA. An additionally disclosed aspect is automatic state saving where state information is stored automatically during multiple steps in the MUTA; the user then need not select deferral.

WO 01/09714 A2

METHOD, APPARATUS, AND COMPUTER PROGRAM PRODUCT FOR DEFERRED COMPLETION OF MULTI-STEP USER TRANSACTION APPLICATIONS

FIELD

Aspects of the invention relate generally to server architectures for transaction processing and more particularly to architectures with features allowing user control over the timing of the completion portion of transaction processing.

BACKGROUND

The collection of protocols and applications commonly referred to as the World Wide Web ("Web") is a widely used client-server architecture for transaction applications. The convenience of carrying on transactions with Web applications is leading to an increasing portion of the transactions persons engage in day-to-day being performed with Web applications.

Typically, transaction applications involve many steps each with its own page. The network user generally must interact must with each page of the transaction application. When such a multi-step user transaction application (hereinafter "MUTA") is started by the user, they may not have expected the MUTA to require as many steps as it does, or some other activity may interrupt the user's ability to complete the MUTA at the current time. This is particularly true with increasingly popular platforms where, for instance, Web applications, are combined with televised entertainment. Such platforms are frequently known as "Enhanced TV" or "Interactive TV" (hereinafter "ITV"). In such a context, a viewer of a video program may be prompted by a Web resource enhancing the video program to initiate a transaction. However, the user may wish to return to viewing the video program quickly and not wish to spend the time to complete the transaction at that moment.

Thus it would be desirable for a mechanism to exist that allowed the user to defer completion of such a transaction.

Related series of technologies include user registration systems. In a typical user registration system, a user provides information to the operator of, for instance, a Web application. The user may be able to pre-specify information that is relevant to a later interaction during the registration process. However, the registration information that is stored cannot provide information that would allow a user to return to the precise position in a later-initiated transaction where the user decided to defer completion. Put another way, while user data stored, for instance as part of a registration process, may be relevant to a later transaction and form part of the state of the transaction at any particular step, user registration data does not provide full state information for later-initiated transactions. Accordingly, stored registration data technologies are unable to provide a system for preserving state of a MUTA to allow deferred completion and resumption.

An additional aspect of conventional technologies is that conventional Web applications lack a convenient mechanism to defer completion at any point of a transaction. Many conventional electronic commerce applications operate on a 'shopping cart' model. In a typical 'shopping cart' model application, a user will interact with an 'electronic store' of some sort: browsing items, comparing product features, searching, etc., and then add items the user wishes to purchase to an electronic 'shopping cart'. When the user wishes to complete their interaction at the 'electronic store,' the user so indicates and the items 'in' their 'shopping cart' are processed to consummate the transaction. Many 'shopping cart' applications provide the feature that a user can return to their 'shopping cart' with the state preserved, i.e., the items they have previously 'in' their 'shopping cart' will still be there. However, the 'shopping cart' is but one part in an overall shopping application: users do not have the ability to return directly to other points in the overall process, for example a

particular page providing information about a product's features. As the type and nature of Web transactions becomes more complex, and as less sophisticated users carry on such transactions, a need has arisen for a general solution to the problem of allowing a user to defer completion of MUTAs at any step in the overall interaction and resume the interaction at that stage at a later time.

SUMMARY

In order to provide a solution to the forgoing and additional problems, aspects of our invention provide a method, apparatus, and computer program product for deferred completion of multi-step user transaction applications.

One aspect of the invention involves computer-implemented methods for deferring completion of a multi-step user transaction application. An illustrative method includes providing a page to a client application. The page includes one or more resources for input of information related to completion of the MUTA, and a resource for selection of a deferral action. The method also includes receiving an indication of the selection of the deferral action, storing state information (including a position in the MUTA and information related to completion of the MUTA); and generating a resumption object including an identifier of a resource for completion of the MUTA. Storing state information can be performed automatically at more than one step in the MUTA and storing state information can be performed in response to user selection. In addition storing state information may include monitoring an elapsed time since a transaction page was provided; determining whether the elapsed time exceeds a predetermined time limit; and storing state information if a user response is received within the predetermined time limit. In an further step, the resumption object is transmitted to a client application.

An additional variation of the illustrative method includes a computer-implemented method for completing a previously-deferred MUTA. The computer-implemented method includes providing a resumption object to a client application.

The resumption object here includes an identifier of a resource for completion of the MUTA. Additional steps in the method include receiving a request for the resource for completion of the MUTA; retrieving previously-received information for returning to a state at which the MUTA was previously deferred; and providing a resource corresponding to the state at which the MUTA was previously deferred.

Yet another aspect of the present invention are computing apparatuses for deferring completion of a multi-step user transaction application. An illustrative computing apparatus with a processor, a memory, and an input/output system includes server process logic configured for providing a page to a client application, the page including one or more resources for input of information related to completion of the MUTA and a resource for selection of a deferral action. The server process logic is further configured for receiving an indication of the selection of the deferral action, and generating a resumption object comprising an identifier of a resource for completion of the MUTA. The computing apparatus also includes a data storage system configured for storing state information including a position in the MUTA, and information related to completion of the MUTA. In a variation, the server process logic is configured for storing state information in the data storage system at plural steps in the MUTA; and, the illustrative apparatus may inclde process logic configured for monitoring an elapsed time since a transaction page was provided. In yet another component, the server process logic is further configured for transmitting the resumption object to a client application.

An additional variation of the illustrative apparatus includes a data storage system configured for storing a state object and server process logic configured for providing a resumption object to a client application. The resumption object includes an identifier of a resource for completion of the MUTA. The server process logic is also configured to receive a request for the resource for completion of the MUTA, retrieve previously-received information from the state object for returning to a state

at which the MUTA was previously deferred, and provide a resource corresponding to the state at which the MUTA was previously deferred.

Yet another aspect of the invention are computer program products including a computer-readable storage medium having computer-readable program code embodied therein for deferring completion of a multi-step user transaction application. Illustrative computer-readable program code includes code for providing a page to a client application. The page includes one or more resources for input of information related to completion of the MUTA, and a resource for selection of a deferral action. The computer-readable program code also includes code for receiving an indication of the selection of the deferral action, code for storing state information (including a position in the MUTA and information related to completion of the MUTA), and code for generating a resumption object comprising an identifier of a resource for completion of the MUTA. The code for storing state information may also include code for storing state information at two or more steps in the MUTA. The code for storing state information may include code for monitoring an elapsed time since a transaction page was provided; code for determining whether the elapsed time exceeds a predetermined time limit; and code for storing state information if a user response is received within the predetermined time limit.

An additional variation on the illustrative program product includes a computer-readable medium having computer program code embodied therein, the computer program code including code for providing a resumption object to a client application. The resumption object includes an identifier of a resource for completion of the MUTA. The computer program also includes code for receiving a request for said resource for completion of the MUTA, code for retrieving previously-received information for returning to a state at which the MUTA was previously deferred; and code for providing a resource corresponding to said state at which the MUTA was previously deferred.

Another aspect of the invention is that it can obtain additional benefits when implemented in a transaction system that also enables automatic suppression of pages in multi-page forms-based transaction applications. An illustrative computer-controlled method enabling reduction of the number of pages provided for user input in forms-based user transaction applications includes examining a set of form fields associated with information required to complete a user transaction application for determining if sufficient form field values corresponding to the set of form fields are available to complete the user transaction application. If at least one of the form field values corresponding to the set of form fields is unavailable, a page is provided for receiving the at least one of the set of form field values; otherwise a request to a successor page is provided for further processing of the user transaction application. In a variation, this illustrative method includes providing a selectable option for determining whether the user transaction application should have the number of pages for user input reduced and, wherein reducing the number of pages provided for user input in forms-based user transaction applications occurs only if the selectable option is selected.

In another variation of this illustrative method, the set of form fields is divided among a plurality of ordered pages; and the step of providing a request to a successor page includes redirecting a request to the successor page. Still further, providing a page for receiving the at least one of the set of form field values may include: repetitively adding at least one of the form fields corresponding to the at least one of the set of form field values to a developing page; and providing the developing page for receiving the at least one of the set of form field values. In an additional variation, adding at least one of the form fields may include determining if adding at least one of the form fields would result in the developing page, when rendered by a client application, to exceed predetermined limits, and; if not, adding the at least one of the form fields to the developing page. A two-dimensional bin-packing algorithm could be used to arrange form fields on the developing page and determine whether adding a form field would exceed the predetermined limit. The predetermined limit may be

based, for instance, on the screen size (in pixels) of a television. The size of form fields can also be denominated in pixels and used by the bin-packing algorithm.

Additionally, computing apparatus for deferring completion of a multi-step user transaction application can be further configured with programmed instructions for reducing the number of pages provided for user input in forms-based user transaction applications. Such programmed instructions configure the computing apparatus to provide structures for implementing particular functions in accordance with some embodiments of the invention. In an illustrative computing apparatus also enabling reducing the number of pages provided for user input in forms-based user transaction applications, the programmed instructions configure the computing apparatus to provide structures for: examining a set of form fields associated with information required to complete a user transaction application; determining if sufficient form field values corresponding to the set of form fields are available to complete the user transaction application; and providing a page for receiving the at least one of the set of form field values if at least one of the form field values corresponding to the set of form fields is unavailable and otherwise providing a request to a successor page for further processing of the user transaction application. In a variation, the illustrative method includes providing a selectable option for determining whether said user transaction application should have the number of pages for user input reduced and, wherein reducing the number of pages provided for user input in forms-based user transaction applications occurs only if the selectable option is selected. In a variation the set of form fields is divided among a plurality of ordered pages; and providing a request to a successor page includes redirecting a request to the successor page. In addition, providing a page for receiving the at least one of the set of form field values may include repetitively adding at least one of the form fields corresponding to the at least one of the set of form field values to a developing page; and providing the developing page for receiving the at least one of the set of form field values. In a variation, adding at least one of the form fields may include determining if adding at least one of the form fields would result in the

developing page, when rendered by a client application, to exceed predetermined limits, and if not, adding the at least one of the form fields to the developing page.

A computer readable storage media having computer readable code embodied therein for for deferring completion of a MUTA can further include code for reducing the number of pages provided for user input in forms-based user transaction applications. An Illustrative storage medium for reducing the number of pages provided for user input in forms-based user transaction applications further includes code for examining a set of form fields associated with information required to complete a user transaction application; code for determining if sufficient form field values corresponding to the set of form fields are available to complete the user transaction application; and code for providing a page for receiving the at least one of the set of form field values if at least one of the form field values corresponding to the set of form fields is unavailable. Otherwise, a request to a successor page is provided for further processing of the user transaction application. In another aspect, the code providing a page for receiving the at least one of the set of form field values may include: code for repetitively adding at least one of the form fields corresponding to the at least one of the set of form field values to a developing page; and code for providing the developing page for receiving the at least one of the set of form field values. Also, adding at least one of the form fields may include determining if adding at least one of the form fields would result in the developing page, when rendered by a client application, to exceed predetermined limits, and if not, adding the at least one of the form fields to the developing page. In another variation the set of form fields is divided among a plurality of ordered pages; and providing a request to a successor page comprises redirecting a request to the successor page.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features, aspects, and advantages of the present invention will become better understood with regard to the following description, appended claims, and accompanying drawings where:

Fig. 1 is a diagram of elements in an operating environment in which an illustrative embodiment can be employed;

Fig. 2 depicts a computer system capable of being configured to embody aspects of the invention in accordance with an illustrative embodiment;

Fig. 3 is a state diagram showing deferred completion of a MUTA in accordance with an illustrative embodiment;

Fig. 4 is a diagram of a completion deferral data architecture in accordance with an illustrative embodiment;

Fig. 5 depicts a rendered screen display of a step in an illustrative MUTA ;

Fig. 6 depicts a flow diagram of a method for deferring completion of a MUTA in accordance with an illustrative embodiment;

Fig. 7 depicts a flow diagram of a method for deferring completion of a MUTA with automatic state saving in accordance with an illustrative embodiment; and

Fig. 8 depicts a flow diagram of a method for completing a previously-deferred MUTA in accordance with an illustrative embodiment.

DETAILED DESCRIPTION

DESCRIPTION OF FIGURES

Fig. 1 depicts elements in an operating environment **1000** in accordance with an illustrative embodiment. Shown are a client application **1100** and a server process **1300**. The client application **1100** and the server process **1300** are configured for client-server communication across a data network **1200**. In some embodiments, the data network **1200** comprises a portion of the Internet, although other networks could be used, either public or private, and using either the TCP/IP protocols (including the User Datagram Protocol) or with other protocols. In preferred embodiments the Hypertext Transfer Protocol ("HTTP") is used to communicate Request and Response messages and Hypertext Markup Language ("HTML") pages.

In some embodiments, the client application **1100** executes on a hardware platform that integrates the features of a television receiver and network connectivity. Many commercially available structures can perform these functions. Certain of the current generation of "set-top boxes" are suitable, including, for instance, the WebTV "Plus" set-top box (also known as an Internet Receiver available through Microsoft's WebTV Networks, Inc. of Palo Alto, California (and their manufacturing licensees). In other embodiments, the client application could execute on a general purpose computer configured with suitable video hardware to integrate the features of a television receiver and network connectivity. As one skilled in the art will appreciate from this disclosure, the features of the invention are not limited to embodiments which operate with client platforms that integrate television and network connectivity. However, the features of the invention obtain particular benefits in this context.

Fig. 2 depicts a computer system **2000** capable of embodying aspects of the invention. The server process **1300** may execute on structures in accordance with the computer system **2000**. The computer system **2000** comprises a microprocessor **2010**, a memory **2020** and an input/output system **2030**. The memory **2020** is capable of

being configured to provide a data structure **2040** which may contain data manipulated by the computer system **2000** when embodying aspects of the invention. Further illustrated is a media drive **2070**, such as a disk drive, CD-ROM drive, or the like. The media drive **2070** may operate with a computer-usable storage medium **2075** capable of storing computer-readable program code ("code") able to configure the computer system **2000** to embody aspects of the invention. The input/output system **2030** may also operate with a keyboard **2050**, a display **2060**, a pointing device **2090**, or a network such as the data network **1200**. The input/output system **2030** may also communicate with a mass data storage **2080** such as a database or the like. The mass data storage **2080** may store information related to users of the client application **1100** that facilitate completion of MUTA, including, for instance, user profile information, records of past interactions, payment instrument information, demographic information, product or service feature preference information, or the like.

As illustrated, the computer system **2000** is general-purpose computing machinery. As one of skill recognizes, programmed instructions may configure general purpose computing machinery to embody structures capable of performing functions in accordance with aspects of the invention. Special purpose computing machinery comprising, for example, an application specific integrated circuit (ASIC) may also be used. In addition, the computer system **2000** could be diskless and operate with a non-volatile memory. Further, configurable hardware could be used including, for instance, a field programmable gate array, or a complex programmable logic device. One skilled in the art will recognize, numerous structures of programmed or programmable logic capable of being configured to embody aspects of the invention. In some embodiments, the computer system **2000** is a SPARC-based workstation from Sun Microsystems of Mountain View, CA, running the SOLARIS operating system and the Apache HTTP server with a Secure Sockets Layer module. In such embodiments, the server process **1300** could comprise the Apache HTTP server with the Secure Sockets Layer module.

The client application **1100** (under the direction of a user) may communicate with the server process **1300** to carry out a MUTA. The user transaction application may be a commercial transaction (sometimes referred to as "E-Commerce"), although one skilled in the art is familiar with many other user transaction applications conventionally practiced in, for instance the Internet environment.

Typically MUTAs involve many steps, each with its own page(s). The user may not have expected so many steps when initiating the MUTA, or some other activity may interrupt the user's ability to complete the application at the current time. This is particularly true of ITV combinations of World Wide Web applications and televised entertainment where the user may wish to return to watching television full-screen before the interactive aspects of the application are completed.

Fig. 3 depicts a deferred completion state diagram **3000** illustrative of various embodiments of the invention. The deferred completion state diagram **3000** is a high level representation of the context in which features of the invention may be advantageously employed. As depicted in **Fig. 3**, states and transitions of a MUTA are shown. When a user initiates the MUTA a first state **3100** is entered. The first state **3100** may be understood on differing levels of generality, from the first page provided from the server process **1300** to the client application **1100**, to greater levels of generality including parameters relevant to the MUTA and/or their values. Without deferral, the MUTA proceeds though a first to second state transition **3800** to a second state **3200**, though a second to third state transition **3850** to a third state **3300**, etc. until a last or Nth state **3400** is entered and the MUTA completes. For similar transactions, one ordinarily expects the number of pages of forms in a transaction to be greater when the client platform is an enhanced or interactive television device than when the client platform is a conventional PC-based browser. The number of states at which a transaction could be deferred would typically grow along with the number of pages of forms in the transaction.

Fig. 3 also depicts a deferred state **3500** entered through a deferral transition **3600**. For instance, if the second state **3200** were the second page of the MUTA, the user could elect to defer completion of the MUTA and transition from the second state **3200** to the deferred state **3500** with the deferral transition **3600**. Once deferred, the MUTA remains in the deferred state **3500** until the user decides to complete it. When the user decided to complete it, a resumption transition into the second state **3720** would return the MUTA to the second state **3200**--the position from which the user entered the deferred state **3500**. **Fig. 3** illustrates this with a parenthetical notation of the resumption transition given the position at deferral, e.g. (R|P=2) for "Resume given that position at deferral was state 2". The user could then proceed to complete the MUTA from the second state **3200**. **Fig. 3** also shows a resumption transition into the third state **3730** and a resumption transition into the Nth state **3740** indicating that the features of the invention provide for deferral and resumption from any state of the MUTA.

Fig. 4 depicts an data architecture **4000** in accordance with an illustrative embodiment. In this illustrative embodiment, the server process **1300** configures the memory **2020** of the computer system **2000** to provide the data structure **2040** in accordance with the data architecture **4000** for deferral and resumption of MUTAs. In other embodiments, information stored in the data architecture **4000** could be stored in the mass data storage **2080** and maintained with a conventional database management system. The data architecture **4000** comprises, for instance a table or tree, keyed by a user identifier **4100**. The user identifier **4100** could be an identifier of a user associated generally with an account maintained by an operator of the server process **1300**, could be established on a per-transaction basis, could be a more general form of personal identification information such as a PIN number, or other identifier. **Fig. 4** also depicts a first transaction state object **4200** comprising a transaction identifier **4210**, a position identifier **4220**, and a set of field/value pairs **4230**. State objects such as the first transaction state object **4200** are created in accordance with the illustrative embodiment when deferred completion of the MUTA is elected by the user.

The transaction identifier **4210** is an identifier of a particular transaction for which deferred completion has been elected. A user may defer completion of plural transactions, as illustrated by a second transaction state object **4300** and a Nth transaction state object **4400**. The transaction identifier **4210** in the first transaction state object **4200** distinguishes the transaction associated with the first transaction state object **4200** from those associated with the second transaction state object **4300** or the Nth transaction state object **4400**.

The position identifier **4220** in the first transaction state object **4200** identifies a position in the flow of steps the MUTA comprises to which the user should be returned when resuming the MUTA. The position could be the position at which the user elected deferred completion; the position could also be context-sensitive and, for instance, return the user to a contextually-appropriate prior step. When, for instance, the MUTA is implemented through one or more text markup language documents, e.g. HTML pages, the first transaction state object **4200** may conveniently be a URI.

The set of field/value pairs **4230** in the first transaction state object **4200** stores data fields and associated data field values relevant to completion of the MUTA that have been made available up to the point at which completion deferral was elected. This may include, for instance, user data input as part of the MUTA, data from the user from previous transactions, data obtained from other sources, e.g., user profile information, accounting information internal to the operator of the server process **1300**, etc.

In some embodiments, the MUTA is implemented as a plurality of text markup language documents, possibly including dynamic contents through server-side scripts or programs, and/or client-side scripts or programs. One illustrative embodiment implements the MUTA as a plurality of JavaServer Pages which dynamically produce HTML-based or JavaScript/ECMAScript-based forms for rendering by the client application **1100**.

Fig. 5 depicts a rendered screen display of a step in the MUTA in accordance with this illustrative embodiment. The HTML or JavaScript/ECMAScript forms provide resources for the input of information related to completion of the MUTA. The forms may be rendered by the client application **1100** as illustrated by a first form field **5100**, a second form field **5200**, and a third form field **5300** in **Fig. 5**. In addition, a resource for selection of a deferral action is shown by a 'defer' button **5400**. The resource for selection of a deferral action may also be implemented in HTML, a scripting language, or using other Internet programming techniques known in the art. The particular manner of implementation is not fundamental.

Again making reference, to **Fig. 3**, the MUTA can be considered to be in, for instance the second state **3200**, when the rendered display of **Fig. 5** is being presented to the user. If the user were to select the deferral action by using a selection device to select the 'defer' button **5400**, the deferral transition **3600** would be followed to enter the deferred state **3500**.

To further illustrate features of deferred completion of MUTAs, reference can be made to **Fig. 6** where a flow diagram of a 'deferred completion' method **6000** in accordance with an illustrative embodiment is shown. The 'deferred completion' method **6000** may be carried out by the server process **1300** when engaging in client server communications with the client application **1100**. As depicted in **Fig. 6**, the 'deferred completion' method **6000** is shown initiating at an intermediate point in a MUTA. As one of skill in the art will appreciate, the particular steps which come before the 'deferred completion' method **6000** in the MUTA are not fundamental and the completion deferral features illustrated by the 'deferred completion' method **6000** could be added to conventional transacting systems.

Process flow initiates at a 'start' terminal **6100** and continues to a 'generate next transaction page' process **6200**. The 'generate next transaction page' process **6200** involves the server process **1300** creating the next in a series of pages

implementing the information flow of the MUTA. This may occur in any conventional manner. In preferred versions of the invention, a server-side programming architecture, such as JavaServer Pages, is used and pages with application logic interact with the server process **1300** to create the contents of the next page dynamically.

Process flow continues to a 'provide next transaction page' process **6300** in which the server process **1300** transmits the next page from the 'generate next transaction page' process **6200** to the client application **1100**. The client application **1100** parses and renders the next page and awaits user interaction. As noted in connection with **Fig. 5**, the user may enter data in form fields or interact with information interchange aspects of the MUTA. In addition, the user may elect deferred completion of the MUTA. The user then responds to the next page; the server process **1300** receives a 'user response' data block **6350**, and process flow continues to a 'deferral' decision process **6400**.

The 'deferral' decision process **6400** examines the 'user response' data block **6350** to determine if the user has elected deferred completion. If so, process flow continues to an 'state object setup' process **6500** which creates a state object similar to, for instance the first transaction state object **4200**. The 'state object setup' process **6500** then stores in the state object a transaction identifier, data fields and values necessary for completion of the MUTA, and an identifier of the current position in the MUTA. The data fields and values stored in the state object preferably comprise all data fields and values necessary to complete the MUTA that have been entered by the user up to this point. In addition if sufficient values for data fields that are necessary to complete the MUTA are available without user input, for instance from a record of a previous interaction, these data fields and values are also stored in the state object.

Process flow continues to a 'provide resumption object' process **6600** that creates and provides to the user an object that will allow the user to resume the

MUTA they have deferred. In some embodiments, the resumption object is a URI that identifies a resumption resource. When a request is made to the resumption resource, the data fields and values stored in the state objected would be retrieved and the request is redirected to the position stored in the state object. One skilled in the art will appreciate many other similar implementations where the redirection is to a URI functionally equivalent as to the position stored in the state object. In some embodiments, the URI that identifies the resumption resource could be emailed to the user.

In other embodiments, the resumption resource is the resource referenced by the position stored in the state object. In these embodiments, the resumption object could then be the resumption resource. The resumption object could be provided to the user through email or other messaging system.

In still other embodiments, the server process **1300** could be configured to provide a page showing a list of the transactions the user has deferred and a resumption resource for each of the list of transactions. The resumption object could be a URI that is emailed to the user that would provide the page when the user's client application requested the URI. In a variation, the server process could generate the page and the page could be emailed to the user.

Numerous other variations will be apparent to one skilled in the art that are within the scope and spirit of the invention. From the 'provide resumption object' process **6600**, process flow completes through an 'end' terminal **6800**

If the user does not elect to defer completion, the 'deferral' decision process **6400** exits through its 'no' branch and process flow continues to a 'complete' decision process **6700** that determines if steps remain in the MUTA. If so, the 'complete' decision process **6700** exits through its 'yes' branch and process flow returns to the 'generate next transaction page' process **6200** for another iteration.

When no steps remain in the MUTA, the 'complete' decision process **6700** exits through its 'yes' branch and process flow completes through the 'end' terminal **6800**.

As described with reference to **Fig. 5** and **Fig. 6**, when the user selects to defer completion through a selectable deferral action resource, for instance the 'defer' button **5400**, a state object was created to store the state of the MUTA so the user could later complete it. In other variations, a state object is maintained throughout the MUTA so that that user need not affirmatively elect to defer completion. Rather, if the user simply fails to complete the MUTA for any reason, it may be completed by resuming at a later time.

Fig. 7 depicts a flow diagram of a 'deferred completion with automatic state saving' method **7000** in accordance with an illustrative embodiment. In **Fig. 7** initial steps of a MUTA (which were omitted in **Fig. 6**) are shown. Process flow initiates at a 'start' terminal **7100** and continues when the server process **1300** receives an 'initiating request' data block **7200** from the client application **1100** of a user. The 'initiating request' data block **7200** in preferred embodiments is an HTTP Request Message for a resource which initiates a MUTA.

Process flow continues to a 'state object setup' process **7300** that creates a state object such as the first transaction state object **4200** and sets a transaction identifier, a position identifier, and any data field/value pairs that may be available. Processing continues to a 'generate next transaction page' process **7400** and a 'provide next transaction page' process **7500** that executes analogously to the corresponding processes described in connection with **Fig. 6**.

Next, a timeout timer **7600** executing on the server process **1300** awaits a response from the client application **1100**. If no response arrives within a predetermined time period, the timeout timer **7600** exits through its 'automatic deferral' branch and process flow continues to a 'provide resumption object' process

7850 that executes analogously to the corresponding processes described in connection with Fig. 6. Process flow then completes through an 'end' terminal 7950. In some embodiments, the timeout timer 7600 (or other process) could also be configured to insure that the user's state is saved should an exceptional event force interruption of the MUTA session. For instance, should the server process 1300 crash or otherwise fail in a manner forcing interruption of the session with the client application 1100, rather than simply exit, the state of the MUTA could then be saved enabling the user to resume from that point.

If a response is received from the client application 1100 within the predetermined time period, the timeout timer 7600 exits through its 'timely response' branch and data from a 'user response' data block 7700 is processed. Next a 'state object update' process 7800 updates the state object created by the 'state object setup' process 7300: the position identifier is altered to reflect that the MUTA has advanced a step, and data received in the 'user response' data block 7700 is added to the data fields/values. Process flow continues to a 'complete' decision process 7900 that executes analogously to the corresponding processes described in connection with Fig. 6 and process flow either returns to the 'generate next transaction page' process 7400 for another iteration or completes through the 'end' terminal 7950.

A user should be able to resume a previously deferred MUTA to complete it. Fig. 8 depicts a flow diagram of a 'MUTA resumption' method 8000 in accordance with an illustrative embodiment. The 'MUTA resumption' method 8000 is carried out by the server process 1300. Process flow initiates at a 'start' terminal 8100 and continues to receive a 'resumption URI request' data block 8200 from the client application 1100 of the user.

As described with reference to the 'provide resumption object' process 6600, the user is provided an identifier of a resumption resource in a resumption object. In this illustrative embodiment, the resumption object comprises a resumption resource

in the form of a URI in the 'resumption URI request' data block **8200**. In preferred embodiments the 'resumption URI request' data block **8200** is an HTTP Request Message and the server process **1300** parses the Request Message; extracts the resumption URI; and maps the resumption URI to an internal resource used to access the state object associated with the MUTA. For instance, the mapping may be to a table lookup operation or database query depending, for instance, on the storage architecture for state objects.

Process flow continues to a 'retrieve position' process **8300** and a 'retrieve fields/values' process **8400**. In the 'retrieve position' process **8300**, the server process **1300** accesses the state object and retrieves the position identifier from the state object. Similarly, in the 'retrieve fields/values' process **8400**, the server process **1300** retrieves the data field/value pairs from the state object.

During the period between when a MUTA is deferred and when it is resumed, additional information required to complete the MUTA may have been entered by the user or otherwise made available to the server process **1300**. The additional information could be stored in a user profile. When the user resumes the MUTA after deferral, a 'merge profile information' process **8500** examines the user profile to determine what information in the user profile could be used in completion of the remainder of the user transaction application.

Process flow continues to a 'redirect to position' process **8700** process where the server process **1300** redirects the Request Message from the 'resumption URI request' data block **8200** to the position identifier retrieved from the state object by the 'retrieve position' process **8300**. Next, a 'transaction completion' process **8800** allows the user to complete the MUTA. The 'transaction completion' process **8800** may, for instance, be to return to processing such as that described above in connection with **Fig. 6**. Process flow completes through a 'end' terminal **8900**.

Additional disclosure may be found in the pages identified as Appendix A which form a portion of this Specification.

Although the present invention has been described in terms of features illustrative embodiments, one skilled in the art will understand that various modifications and alterations may be made without departing from the scope of the invention. Accordingly, the scope of the invention is not to be limited to the particular embodiments discussed herein, but should be defined only by the allowed claims and equivalents thereof.

APPENDIX A

METHOD, APPARATUS, AND COMPUTER PROGRAM PRODUCT FOR AUTOMATIC PAGE SUPPRESSION IN FORMS

FIELD

The invention relates generally systems for on-line transactions and, more particularly, to transaction architectures in client server communications employing markup language based electronic forms for processing transaction related information.

BACKGROUND

Client-server applications employing markup language based electronic documents, e.g. the World Wide Web, often involve forms (also referred to herein as "forms pages"). Electronic commerce applications are particularly reliant on forms for receiving information from a party to a transaction that information which is necessary to complete the transaction. Especially as these applications are made available to devices with limited resolutions (compared to the displays commonly used with conventional personal computers or workstations) these forms are spanning multiple pages. This trend towards the proliferation of forms pages has at least two undesirable effects.

A first undesirable effect is that a succession of pages can become tedious and frustrating to the user. Users are frequently drawn to, for instance World Wide Web applications, because of their increased convenience over conventional methods of transacting. A tedious succession of forms pages to complete a transaction negates at least a portion of the user benefits to transacting in this way. "Impulse transactions" are desirable from a consumer perspective, as they allow the consumer to immediately (or more nearly so) obtain and enjoy items they wish to consume and the tedious

APPENDIX A (continued)

succession of forms pages may prevent "impulse transactions." Accordingly, it would be desirable for a system to exist that allowed for the reduction or elimination of the number of forms pages required for a user to complete a forms-based transaction application.

A second undesirable effect is that multiple forms pages involve multiple requests for resources of the server, thereby increasing server overhead. For instance, an operator of an electronic marketplace may have an appreciable amount of overhead costs devoted to hardware for adequate transaction processing abilities to serve customers promptly. If each transaction requires processing multiple forms pages, as the number of forms pages increases, the operator needs to invest in additional computing hardware to maintain the same level of prompt customer service. Thus it would be advantageous for means to exist by which server operators processing forms-based transactions could reduce the number of forms pages required to complete a transaction.

Increasingly, user transaction applications cultivate repeat usage through a user registration process or by storing for later use the information provided by users in previous transaction sessions. In either instance, a sever process carrying on a user transaction application may have available to it sufficient information to complete, at least some, of the form fields that are required to complete the user transaction application. Further, a repeat user may desire that the server process use the available sufficient information without confirmation to complete the user transaction application. In particular, for forms pages that contain only fields for which sufficient form field values are already available, given the user's consent it may be desirable to suppress providing those forms pages to the user.

One conventional approach directed towards streamlining user transaction applications are conventional "one-click" or "single step" ordering systems used in electronic commerce transactions. An exemplary system is described in WO9913424,

APPENDIX A (continued)

entitled "Method and System for Placing a Purchase Order Via A Communications Network" for applicant Amazon.com. Single-step ordering systems do streamline product purchase transactions, however they do not provide a solution for automatic page suppression in forms. Typically "single-step" systems involve a pre-registration process in which a user enters the information necessary to complete transactions (generally spanning multiple forms pages) with the system. Then, when a subsequent transaction is initiated, either the entire series of transaction pages is skipped or the entire series of transaction pages must have their form field values re-entered by the user. That is, conventional systems provide no logic with respect to which aspects of a series of forms pages can be suppressed.

Conventional systems present significant disadvantages if the user is engaged in activity in addition to completing the user transaction application. For instance, when a user's client application is a device that integrates television reception with, for instance HTTP client capability, the user may initiate a user transaction application in response to an interactive portion of a video program. In this context, the user could naturally desire to complete the user transaction application with dispatch so that they can return to viewing the video program. Conventional single-step systems would be inadequate solutions in this context. This is due, in part, to the fact that they typically require the user to process the entire series of forms pages to complete input of only a few form fields, when sufficient form field values are unavailable for the few form fields. Accordingly, there is a need for a system of automatic page suppression in forms that is flexible and optimized to suppress as much of a series of forms pages as can be suppressed with user consent.

SUMMARY

In order to provide a solution to the forgoing and additional problems, aspects the present invention provide a method, apparatus, and system for automatic page suppression in forms.

APPENDIX A (continued)

One aspect of the invention involve methods for reducing the number of pages provided for user input in forms-based user transaction applications. An illustrative method includes examining a set of form fields associated with information required to complete a user transaction application for determining if sufficient form field values corresponding to the set of form fields are available to complete the user transaction application; and if at least one of the form field values corresponding to the set of form fields is unavailable, providing a page for receiving the at least one of the set of form field values; and otherwise providing a request to a successor page for further processing of the user transaction application.

In a variation, the illustrative method includes providing a selectable option for determining whether the user transaction application should have the number of pages for user input reduced and, wherein reducing the number of pages provided for user input in forms-based user transaction applications occurs only if the selectable option is selected.

In another variation, the set of form fields is divided among a plurality of ordered pages; and the step of providing a request to a successor page comprises redirecting a request to the successor page. Still further, providing a page for receiving the at least one of the set of form field values may include: repetitively adding at least one of the form fields corresponding to the at least one of the set of form field values to a developing page; and providing the developing page for receiving the at least one of the set of form field values. In an additional variation, adding at least one of the form fields may include determining if adding at least one of the form fields would result in the developing page, when rendered by a client application, to exceed predetermined limits, and; if not, adding the at least one of the form fields to the developing page.

Yet another aspect of the invention is a computing apparatus configured with programmed instructions for reducing the number of pages provided for user input in

APPENDIX A (continued)

forms-based user transaction applications. The programmed instructions configure the computing apparatus to provide structures for implementing particular functions in accordance with some embodiments of the invention. In an illustrative computing apparatus, the programmed instructions configure the computing apparatus to provide structures for: examining a set of form fields associated with information required to complete a user transaction application; determining if sufficient form field values corresponding to the set of form fields are available to complete the user transaction application; and providing a page for receiving the at least one of the set of form field values if at least one of the form field values corresponding to the set of form fields is unavailable and, otherwise providing a request to a successor page for further processing of the user transaction application.

A variation includes providing a selectable option for determining whether the user transaction application should have the number of pages for user input reduced and, wherein reducing the number of pages provided for user input in forms-based user transaction applications occurs only if the selectable option is selected.

In a variation the set of form fields is divided among a plurality of ordered pages; and providing a request to a successor page includes redirecting a request to the successor page. In addition, providing a page for receiving the at least one of the set of form field values may include repetitively adding at least one of the form fields corresponding to the at least one of the set of form field values to a developing page; and providing the developing page for receiving the at least one of the set of form field values.

In another variation, adding at least one of the form fields may include determining if adding at least one of the form fields would result in the developing page, when rendered by a client application, to exceed predetermined limits, and if not, adding the at least one of the form fields to the developing page.

APPENDIX A (continued)

Yet another aspect of the invention is a computer program product comprising a computer readable storage medium having computer readable code embodied therein for reducing the number of pages provided for user input in forms-based user transaction applications. The computer readable code includes code for examining a set of form fields associated with information required to complete a user transaction application; code for determining if sufficient form field values corresponding to the set of form fields are available to complete the user transaction application; and code for providing a page for receiving the at least one of the set of form field values if at least one of the form field values corresponding to the set of form fields is unavailable and otherwise providing a request to a successor page for further processing of the user transaction application.

In another aspect, the code providing a page for receiving the at least one of the set of form field values may include: code for repetitively adding at least one of the form fields corresponding to the at least one of the set of form field values to a developing page; and code for providing the developing page for receiving the at least one of the set of form field values. Also, adding at least one of the form fields may include determining if adding at least one of the form fields would result in the developing page, when rendered by a client application, to exceed predetermined limits, and if not, adding the at least one of the form fields to the developing page. In another variation the set of form fields is divided among a plurality of ordered pages; and providing a request to a successor page comprises redirecting a request to the successor page.

Another aspect of the invention is that it can obtain additional benefits when implemented in a transaction system that also enables deferring completion of a multi-step user transaction application. Illustratively, compter-implemented methods enabling this feature include providing a page to a client application. The page includes one or more resources for input of information related to completion of the multi-step user transaction application ("MUTA"), and a resource for selection of a

APPENDIX A (continued)

deferral action. This method also includes receiving an indication of the selection of the deferral action, storing state information (including a position in the MUTA and information related to completion of the MUTA); and generating a resumption object including an identifier of a resource for completion of the MUTA.

The state information could be stored in a conventional data structure. Storing state information can be performed automatically at more than one step in the MUTA and storing state information can be performed in response to user selection. In addition storing state information may include monitoring an elapsed time since a transaction page was provided; determining whether the elapsed time exceeds a predetermined time limit; and storing state information if a user response is received within the predetermined time limit. In an further step, the resumption object is transmitted to a client application. In some instances, the resumption object includes a URL or other resouce reference.

An additional feature of this illustrative method enabling deferred completion includes a computer-implemented method for completing a previously-deferred MUTA. This method includes providing a resumption object to a client application. The resumption object here includes an identifier of a resource for completion of the MUTA. Additional steps in the method include receiving a request for the resource for completion of the MUTA; retrieving previously-received information for returning to a state at which the MUTA was previously deferred; and providing a resource corresponding to the state at which the MUTA was previously deferred.

A computing apparatus configured with programmed instructions for reducing the number of pages provided for user input in forms-based user transaction applications can further be configured for enabling deferred completion of MUTAs. An illustrative computing apparatus also includes server process logic configured with programmed instructions for providing a page to a client application, the page including one or more resources for input of information related to completion of the

APPENDIX A (continued)

MUTA and a resource for selection of a deferral action. The apparatus is further configured for receiving an indication of the selection of the deferral action, and for generating a resumption object. The resumption object includs an identifier of a resource for completion of the MUTA. The computing apparatus also includes a data storage system configured for storing state information including a position in the MUTA, and information related to completion of the MUTA. In a variation, the apparatus is configured for storing state information in the data storage system at plural steps in the MUTA; and, the illustrative apparatus may include timing process logic configured for monitoring an elapsed time since a transaction page was provided. In yet another component, the apparatus is further configured for transmitting the resumption object to a client application.

An additional variation of the illustrative apparatus described above includes a data storage system configured for storing a state object and server process logic configured for providing a resumption object to a client application. The resumption object includes an identifier of a resource for completion of the MUTA. The server process logic is also configured to receive a request for the resource for completion of the MUTA, retrieve previously-received information from the state object for returning to a state at which the MUTA was previously deferred, and provide a resource corresponding to the state at which the MUTA was previously deferred.

Computer readable storage media having computer readable code embodied therein for reducing the number of pages provided for user input in forms-based user transaction applications can further include code for deferring completion of a MUTA. Illustrative computer-readable program code for deferring completion of a MUTA includes code for providing a page to a client application. The page includes one or more resources for input of information related to completion of the MUTA, and a resource for selection of a deferral action. The computer-readable program code also includes code for receiving an indication of the selection of the deferral action, code for storing state information (including a position in the MUTA and information

APPENDIX A (continued)

related to completion of the MUTA), and code for generating a resumption object comprising an identifier of a resource for completion of the MUTA. The code for storing state information may also include code for storing state information and two or more steps in the MUTA. The code for storing state information may include code for monitoring an elapsed time since a transaction page was provided; code for determining whether the elapsed time exceeds a predetermined time limit; and code for storing state information if a user response is received within the predetermined time limit.

An additional variation on the illustrative computer program product involves the computer program code including code for providing a resumption object to a client application. The resumption object includes an identifier of a resource for completion of the MUTA. The computer program product also includes code for receiving a request for said resource for completion of the MUTA, code for retrieving previously-received information for returning to a state at which the MUTA was previously deferred; and code for providing a resource corresponding to said state at which the MUTA was previously deferred.

BRIEF DESCRIPTION OF THE DRAWINGS

These and other features, aspects, and advantages of the present invention will become better understood with regard to the following description, appended claims, and accompanying drawings where:

Fig. A1 is a diagram of elements in an operating environment in which an illustrative embodiment can be employed;

Fig. A2 depicts a computer system capable of being configured to embody aspects of the invention in accordance with an illustrative embodiment;

APPENDIX A (continued)

Fig. A3 is a diagram of a conventional user transaction application using form pages;

Fig. A4 depicts a flow diagram of a method for automatic page suppression in forms in accordance with an illustrative embodiment; and

Fig. A5 depicts a flow diagram of a method for automatic page suppression in forms with just in time pagination.

DETAILED DESCRIPTION

DESCRIPTION OF FIGURES

Fig. A1 depicts elements in an operating environment **1000** in accordance with an illustrative embodiment. A client application **1100** and a sever process **1300** are shown. The client application **1100** and the sever process **1300** are configured for client-server communication across a data network **1200**. In some embodiments, the data network **1200** comprises a portion of the Internet, although other networks could be used, either public or private, using either the TCP/IP protocols (including the User Datagram Protocol) or with other protocols. In preferred embodiments the Hypertext Transfer Protocol ("HTTP") is used to communicate Request and Response messages and Hypertext Markup Language ("HTML") pages. Other embodiments could use the Extensible Markup Language ("XML") or other markup language.

In some embodiments, the client application **1100** executes on a hardware platform that integrates the features of a television receiver and an HTTP client. Many commercially available structures can perform these functions. Certain of the current generation of "set-top boxes" are suitable, including, for instance, the WebTV "Plus" set-top box (also referred to as an Internet Receiver) available through Microsoft's WebTV Networks, Inc. of Palo Alto, California (and its manufacturing licensees). In other embodiments, the client application could execute on a general

APPENDIX A (continued)

purpose computer configured with suitable video hardware to integrate the features of a television receiver and a client application. As one skilled in the art will appreciate from this disclosure, the features of the invention are not limited to embodiments which operate with client platforms that integrate television and network connectivity. However, the features of the invention obtain particular benefits in this context.

Fig. A2 depicts a computer system **2000** capable of embodying aspects of the invention. The sever process **1300** may execute on structures in accordance with the computer system **2000**. The computer system **2000** comprises a microprocessor **2010**, a memory **2020** and an input/output system **2030**. The memory **2020** is capable of being configured to provide a data structure **2040** which may contain data manipulated by the computer system **2000** when embodying aspects of the invention. Further illustrated is a media drive **2070**, such as a disk drive, CD-ROM drive, or the like. The media drive **2070** may operate with a computer-usable storage medium **2075** capable of storing computer-readable program code able to configure the computer system **2000** to embody aspects of the invention. In other embodiments, the computer system **2000** could be a diskless computer with a non-volatile memory. The input/output system **2030** may also operate with a keyboard **2050**, a display **2060**, a pointing device **2090**, or a network such as the data network **1200**. The input/output system **2030** may also communicate with a mass data storage **2080** such as a database or the like. The mass data storage **2080** may store information related to users of the client application **1100** that facilitate completion of a user transaction application, including, for instance, user profile information, records of past interactions, payment instrument information, demographic information, product or service feature preference information, or the like.

As illustrated, the computer system **2000** is general-purpose computing machinery. As one of skill recognizes, programmed instructions may configure general purpose computing machinery to embody structures capable of performing functions in accordance with aspects of the invention. Special purpose computing

APPENDIX A (continued)

machinery comprising, for example, an application specific integrated circuit (ASIC) may also be used. Further, configurable hardware could be used including, for instance, a field programmable gate array or a complex programmable logic device. One skilled in the art will recognize, numerous structures of programmed or programmable logic capable of being configured to embody aspects of the invention. In some embodiments, the computer system **2000** is a SPARC-based workstation from Sun Microsystems of Mountain View, CA, running the SOLARIS operating system and the Apache HTTP server with a Secure Sockets Layer module. In such embodiments, the sever process **1300** could comprise the Apache HTTP server with the Secure Sockets Layer module.

Frequently the client application **1100** communicates with the sever process **1300** to carry out a user transaction application. The user transaction application may be a commercial transaction (sometimes referred to as "E-Commerce"), although one skilled in the art is familiar with may other user transaction applications conventionally practiced in, for instance the Internet environment. Frequently, user transaction applications are implemented with "forms." The HTML specification (version 3.2 , the proposed version 4.0, and it is likely later-developed versions available from the World Wide Web Consortium at <http://www.w3c.org> or through the MIT Laboratory for Computer Science in Cambridge, Massachusetts) has particular provisions for forms that are conventionally used for transactions, although many other conventional Internet programming techniques are also used for forms-based user transaction applications.

Fig. A3 depicts a diagram of a conventional user transaction application using form pages. The client application **1100** sends a user transaction application initiation request **3400** to the sever process **1300** to initiate the user transaction application. The user transaction application initiation request **3400** may be a Request Message in accordance with the HTTP specification which comprises a Uniform Resource Identifier ("URI") that corresponds to a resource available on the sever process **1300**

APPENDIX A (continued)

which initiates execution of the user transaction application. Forms are typically used in user transaction applications to receive input from the user that is necessary to complete the user transaction application. A form comprises one or more form fields; the user provides form field values for the one or more form fields. With reference to **Fig. A3**, a first form page **3450** is communicated from the sever process **1300** to the client application **1100** which renders the form page. The client application **1100** then receives user input for the form field values and communicates first form page field values **3500** to the sever process **1300**.

Frequently, and particularly when the client application **1100** renders on a display with limited resolution (compared to a PC) such as a television, forms for completion of the user transaction application span multiple pages. This is depicted in **Fig. A3** where the sever process **1300** communicates a second form page **3550** to the client application **1100** which responds with a second form page field values **3600** after the user has entered the appropriate form field values. This process may continue for several iterations, and a Nth form page **3650** and a Nth form page field values **3700** are depicted for some integer 'N" greater than 2. For PC-oriented forms, N rarely exceeds 5 . However, as noted above, when completion of user transaction applications requires several form pages, server resources may become burdened and user frustration may increase, perhaps reducing the likelihood the user will engage in subsequent transactions with the operator of the sever process **1300**. Accordingly it is desirable that, when possible, form pages for completion of user transaction application should be suppressed.

Fig. A4 depicts a flow diagram of an 'automatic page suppression' method **4000** in accordance with an illustrative embodiment. The 'automatic page suppression' method **4000** is carried on by a server process configured for operation with a dynamic contents document, for instance one that contains application logic that describes how to process a request to generate a response. Several conventional dynamic document architectures are known in the art including, for instance, Sun

APPENDIX A (continued)

Microsystems' JavaServer Pages, or Microsoft Corporation's Active Server Pages. Other conventional server-side scripting architectures could also be used.

The 'automatic page suppression' method **4000** depicts process flow of an interaction of the dynamic contents document and the sever process **1300** in handling a user transaction application initiated from the client application **1100**. Process flow initiates at a 'start' terminal **4050** and continues to receive a 'request' data block **4100** corresponding to an indication from the user of the client application of initiation of the user transaction application. The sever process **1300** retrieves a resource corresponding to the request. In this illustrative embodiment, the resource is a JavaServer Page type dynamic contents document. The server process begins processing the application logic of the dynamic contents document.

One aspect of the application logic is to maintain state information from a first portion of the user transaction application so that it is available to later portions of the user transaction application. For instance, if the user transaction application comprises several pages in which user information input is received, the user information input from the first of the several pages should be maintained for completion of the user transaction application. A 'state object setup' process **4150** executes in which the sever process creates a state object for maintaining state information across plural portions of the user transaction application. When JavaServer Page type dynamic contents documents are used, the state object may be a JAVA Bean. Other state objects could be used depending on the server process architecture, either more sophisticated programatically, or less sophisticated, for instance a temporary file.

Process flow continues to a 'page suppression mode' decision process **4200** that determines if the user transaction application should execute in page suppression mode. A selectable option may be provided for determining whether the user transaction application should execute in page suppression mode. Preferably, when

APPENDIX A (continued)

the selectable option is selected the user transaction application executes with page suppression. In some embodiments, the selectable option is provided to an operator of the server process. In other embodiments, the selectable option is provided to the user of the client application. In this instance, the user selects whether to complete the user transaction application with page suppression and the user's selection is transmitted to the server process. A preliminary page in the user transaction application could provide the selectable option to the user; the selectable option could be a user interface element, for instance, a button, a check box, etc. In still other embodiments, the selectable option could be a field in a profile for the user and the server process could examine this field to determine whether the user transaction application should execute in page suppression mode.

The 'page suppression mode' decision process **4200** is a method executed by the server process in processing the application logic of the dynamic contents document. If the user transaction application should execute in page suppression mode, the 'page suppression mode' decision process **4200** exits through its 'yes' branch. In this illustrative embodiment, the dynamic contents document defines a set of form fields required for completion of the user transaction application. The dynamic contents document further comprises a method invocation for determining whether sufficient form field values are available for the set of form fields so that suppression of the page(s) soliciting user input of the form field values may be suppressed.

When processing the dynamic contents document, the server processes invokes this method and its execution initiates. Process flow continues to a 'required field selection' process **4300**. The 'required field selection' process **4300** selects one of the set of form fields that are required to complete the user transaction application (mnemonically "F"). A user transaction application may have fields that are not required to complete the user transaction application. Preferably the dynamic contents document comprises a set of form fields that are required to complete that portion of

APPENDIX A (continued)

the user transaction application associated with the dynamic contents document and the 'required field selection' process **4300** selects from that set. No particular manner of selection is fundamental, iteration through a list is typically sufficient. Process flow continues to a 'sufficient field value' decision process **4400**.

The 'sufficient field value' decision process **4400** determines if a sufficient form field value exists for the one of set of form fields to complete the user transaction application. The user of the client application submitting the 'request' data block **4100** may have user profile information accessible to the server process that can be accessed to provide the form field value for the one of the set of form fields. Commonly, the user of the client application is queried to provide user identification information, e.g., a user ID and password, at the initiation of a session of which the user transaction application is a part. If this is the case, the user identification information can be used to access the user profile information which can then be examined for the form field value, e.g., a shipping address or postal code for the user, payment instrument information, product size, color, or configuration preferences, etc. In other instances, user profile information is not stored; however, records of previous interactions during the same session could be kept and used for the form field value. For instance, during a first user transaction application a user could purchase a first item; the server process could be configured to store a record of this interaction for the duration of the user's session. Were the user to initiate a second user transaction application to purchase a second item, the stored record of the previous interaction could be accessed by the server for the form field value. Still, further, client-side state objects could also be used for storing the form field value and the client-side state object could be queried from the server process. In addition, a third party information manager (sometimes referred to as an "infomediary") could store the user information and the sever process **1300** could query the third party information manager.

APPENDIX A (continued)

In some embodiments, the determination made by the 'sufficient field value' decision process **4400** as to whether a sufficient form field value is available is context insensitive. For instance, the 'sufficient field value' decision process **4400** may simply determine if there is a non-NULL value for the form field value, and, if so, deem that sufficient information is available with respect to the one of the set of form fields. In other embodiments, the 'sufficient field value' decision process **4400** could perform more complex context-sensitive determinations. Still further, form field value validation logic could also be performed. For instance, a postal code could be verified for an appropriate number of digits, an email address could be examined for compliance with the relevant specifications, a domain name could be resolved to determine if it corresponds to a reachable host, etc.

If the 'sufficient field value' decision process **4400** determines a sufficient form field value is available for the one of the set of form fields, it exits through its 'yes' branch and process flow continues to an 'additional required fields' decision process **4500**. If the 'sufficient field value' decision process **4400** determines that a sufficient form field value is not available, it exits through its 'no' branch. In this instance, the server process completes execution of the invoked method and continues to a 'provide page' process **4700** to provide the dynamic contents document to the client application for receiving the form field value.

When the 'sufficient field value' decision process **4400** exits through its 'yes' branch, process flow continues within the invoked method to the 'additional required fields' decision process **4500**. While members of the set of form fields that are required for completion remain for which the determination has not been made that a sufficient form field value is available, the 'additional required fields' decision process **4500** exits through its 'yes' branch and process flow returns to the 'required field selection' process **4300** for another iteration. When the 'additional required fields' decision process **4500** exits through its 'no' branch, process flow continues to a 'redirection to successor page' process **4600**. The 'additional required fields' decision

APPENDIX A (continued)

process **4500** exits through its 'no' branch only when each of the set of form fields required for completion has been examined and it has been determined that a sufficient form field value is available. That is, the page that would have been provided by the 'provide page' process **4700** did not need to be provided to receive user input of one or more of the form field values: provision of the page has been suppressed.

The 'redirection to successor page' process **4600** is performed by the invoked method and returns a response to the request received in the 'request' data block **4100** that redirects the request to a next page in the user transaction application. As one of skill in the art will appreciate, the redirection may be either entirely internal to the server process, or the client application may be provided with the response and the client application would then submit a next request to the next page. Preferred embodiments employ internal server redirection, however this is not fundamental.

In some embodiments, the 'redirection to successor page' process **4600** appends the set of form fields and their corresponding form field values onto the request when redirecting the request to the next page. This allows information received or retrieved when processing a current page to be made available when processing the next page. Still further, in some applications simply redirecting to the next page's URI circumvents processing of the form values desirable for successful completion of the user transaction application. For instance, processing of a shipping address and postal code could be used to determine, for instance, product sales tax and calculation of the latter could be triggered by the "posting" of the former from its form page. Such applications preferably redirect to the next page's URI while including a query string comprising all required form field/value pairs, this function may be performed by including a formatted query string comprising the set of form fields and their corresponding form field values. From the 'redirection to successor page' process **4600** or the 'provide page' process **4700**, process flow completes through an 'end' terminal **4800**.

APPENDIX A (continued)

JUST IN TIME PAGINATION

In the above-described illustrative embodiment, user information requests were provided though a plurality of ordered pages of predetermined length. However, it is not fundamental that the user transaction application be split across the plurality of ordered pages, or that pages be of a predetermined length. Rather, pagination may occur responsive to the number and type of the set of form fields for which sufficient information is not available. To further illustrate this aspect, **Fig. A5** depicts a flow diagram of a 'just in time automatic page suppression' method **5000** in accordance with an illustrative embodiment. The 'just in time automatic page suppression' method **5000** is similar to the 'automatic page suppression' method **4000** in that it involves the interaction of a server process with a dynamic contents document and shares several steps for performing analogous functions.

Process flow initiates at a 'start' terminal **5005** and continues to receive a 'request' data block **5010** corresponding to an indication from the user of the client application of initiation of the user transaction application. The server process retrieves a resource corresponding to the request. In this illustrative embodiment, the resource is a JavaServer Page type dynamic contents document. The server process begins processing the dynamic contents document. Process flow continues to a 'state object setup' process **5020** that creates a state object for maintaining state information across plural portions of the user transaction application. Next, a 'required field selection' process **5040** executes analogously to the 'required field selection' process **4300** described above. Process flow continues to a 'sufficient field value' decision process **5050** that performs functions similar to both the 'page suppression mode' decision process **4200** and the 'sufficient field value' decision process **4400** of **Fig. A4**.

Just in time pagination effectively provides field suppression on a field-by-field basis and there need not be predetermined pages at all. However, the user or the

APPENDIX A (continued)

operator of the server process may desire to provide all required form fields to the user. For instance, the user may desire to verify or re-enter previously entered form field values. Thus, the features of the 'page suppression mode' decision process **4200** could also be beneficially used with just in time pagination.

A selectable option could be provided to the user or the server process operator as described above in connection with the 'page suppression mode' decision process **4200**. With just in time pagination, preferred embodiments provide all required form fields (with any available form field values filled in for review and editing by the user) if page suppression mode is not selected. The 'sufficient field value' decision process **5050** examines whether page suppression mode has been selected with the selectable option.

If page suppression mode is not selected, the 'sufficient field value' decision process **5050** treats all of the set of form fields as lacking a sufficient form field value and exits through its 'no' branch. When page suppression is selected, the 'sufficient field value' decision process **5050** executes analogously to the 'sufficient field value' decision process **4400** of **Fig. A4**. That is, it exits through its 'no' branch if, and only if, it determines a sufficient form field value is unavailable.

When, the 'sufficient field value' decision process **5050** exits through its 'no' branch the one of the set of form fields ("F") should be added to a page that will be provided to the user for receiving the form field value. When page suppression is not selected, this occurs for each of the set of form fields.

Typically plural form fields may be suitably arranged on a single page. In this illustrative embodiment, the separation of the form fields from the set of form fields for which form field values are required into an appropriate number of pages occurs when the determination is made that a sufficient form field value is not available. The pagination is just in time. In this instance, just in time pagination is accomplished by creating a developing page for a first form field. Additional form fields are added to

APPENDIX A (continued)

the developing page until it is determined that the developing page should be provided to the user to receive user input. If additional form fields require user input a next developing page is created and additional form fields added to it until it is determined that the next developing page should be provided to the user to receive user input. This process can continue until there are no remaining form fields for which user input is required. When the 'sufficient field value' decision process **5050** exits through its 'no' branch, an 'add F to developing page' process **5070** adds the form field to the developing page.

Depending, for instance, on the client application, the number and nature of the set of form fields that are appropriate for a single page may vary. For example, if the client application is a platform integrating television reception and, for instance an HTTP, client, such as a conventional set-top box, then a user's display is likely a conventional television. If the user's display is a conventional television suitable screen area for the page is more limited than if the user's display were a conventional display use with a personal computer or workstation. For instance, suitable screen area of a conventional television may be 544 (width) by 300 (height) pixels. While a single page, when rendered, could exceed this, it may be desirable that a single page fit within the suitable screen area so that the user is not inconvenienced by having to scroll or navigate across multiple screens of the page.

From the 'add F to developing page' process **5070** and when the 'sufficient field value' decision process **5050** exits through its 'yes' branch indicating a sufficient form field value is available, process flow continues to a 'developing page to user' decision process **5060**. The 'developing page to user' decision process **5060** tests two conditions; if either of the conditions are true, the developing page should be provided to the user for data input.

The first test condition is whether the developing page is fulll. A conventional two-dimensional bin packing algorithm may be used for this test, as may simpler

APPENDIX A (continued)

heuristics. If a bin packing algorithm is used, the suitable screen area may be obtained, for instance from a predetermined value used by the server process, or responsive to an inference based on the type of client application. For example, an inference could be made that if the client application is a set-top box, the suitable screen area is that of a television screen. The screen area (in pixels) of a form field along with any previous form fields added to the developing page could be used by the bin packing algorithm to determine if another of the form fields can be added to the developing page. If another form field cannot be added, the developing page is full and the first test condition is true. The second test condition is whether the developing page should be provided even if it is not full. This situation occurs when at least one of the set of form fields has been added to the developing page--but the developing page is not full--and there are no additional required fields. In this instance, the page should be provided even though it is not full and the second test condition is true.

If either the first or second test condition is true, the 'developing page to user' decision process **5060** exits through its 'yes' branch and process flow continues to a 'provide page' process **5080** that provides the completed developing page to the client application for user input of the form field value. When the user input is received, process flow continues to a 'field value(s) storage' process **5090** that stores the form field value received from the user. In some embodiments, the 'field value(s) storage' process **5090** stores the form field value in the state object created by the 'state object setup' process **5020**; in other embodiments, the 'field value(s) storage' process **5090** stores the form field value along with the set of form fields as field/value pairs in a string that can be later passed as a query string portion of a URI as previously described.

As described above, testing of whether the developing page was full occurred after a field was added to the developing page. This is suitable in situations where

APPENDIX A (continued)

form fields are of nearly constant space, e.g. input boxes and drop-down menus. As most user transaction applications are of this type, the post-testing is suitable. In instances where the expected form fields are not of nearly constant space, a pre-addition test could be used and when the developing page was full, a next page started, the form field provided to the next page, and the developing page be provided to the user.

From the 'field value(s) storage' process **5090** and when the 'developing page to user' decision process **5060** exits through its 'no' branch indicating both the first and second test conditions are false, process flow continues to the 'additional required fields' decision process **5100**. The 'additional required fields' decision process **5100** performs analogously to the 'additional required fields' decision process **4500** which was previously described. While members of the set of form fields remain for which the determination has not been made that a sufficient form field value is available, the 'additional required fields' decision process **5100** exits through its 'yes' branch and process flow returns to the 'required field selection' process **5040** for another iteration. When the 'additional required fields' decision process **5100** exits through its 'no' branch, process flow continues to a 'redirection to successor page' process **5130**.

The 'redirection to successor page' process **5130** redirects the request received in the 'request' data block **5010** to the successor page in the user transaction application, which, when just in time pagination is employed, is typically a page that informs the user that the user transaction application has been completed. As described above in connection with the 'redirection to successor page' process **4600**, the redirected request may comprise a query string with field/value pairs. Process flow then completes through an 'end' terminal **5150**.

Although the present invention has been described in terms of features illustrative embodiments, one skilled in the art will understand that various

APPENDIX A (continued)

modifications and alterations may be made without departing from the scope of the invention. Accordingly, the scope of the invention is not to be limited to the particular embodiments discussed herein, but should be defined only by the allowed claims and equivalents thereof.

5

Claims

What is claimed is:

1. A computer-implemented method for deferring completion of a multi-step user transaction application ("MUTA"), said computer-implemented method comprising:

providing a page to a client application, said page comprising

one or more resources for input of information related to completion of said MUTA; and

a resource for selection of a deferral action;

receiving an indication of the selection of said deferral action;

storing state information comprising:

a position in said MUTA; and

information related to completion of said MUTA; and

generating a resumption object comprising an identifier of a resource for completion of said MUTA.

2. The computer-implemented method according to claim 1 wherein said step of storing state information is performed at plural steps in said MUTA.

3. The computer-implemented method according to claim 2 wherein storing state information comprises:

monitoring an elapsed time since a transaction page was provided;

determining whether said elapsed time exceeds a predetermined time limit; and

storing state information if a user response is received within said predetermined time limit.

4. The computer-implemented method according to claim 1 further comprising:

transmitting said resumption object to a client application.

5. An apparatus comprising a processor, a storage, and an input-output system, said apparatus configured for deferring completion of a multi-step user transaction application ("MUTA"), said apparatus comprising:

server process logic configured for:

providing a page to a client application, said page comprising one or more resources for input of information related to completion of said MUTA, and a resource for selection of a deferral action,

receiving an indication of the selection of said deferral action, and

generating a resumption object comprising an identifier of a resource for completion of said MUTA; and

a data storage system configured for storing state information comprising:

a position in said MUTA, and

information related to completion of said MUTA.

6. The apparatus of claim 5 wherein said server process logic is configured for storing state information in said data storage system at plural steps in said MUTA.

7. The apparatus of claim 6 further comprising process logic configured for monitoring an elapsed time since a transaction page was provided.

8. The apparatus of claim 5 wherein said server process logic is further configured for transmitting said resumption object to a client application.

9. A computer program product comprising a computer readable storage medium having computer readable code embodied therein for deferring completion of a multi-step user transaction application, said computer readable code comprising:

code for providing a page to a client application, said page comprising

one or more resources for input of information related to completion of said MUTA; and

a resource for selection of a deferral action;

code for receiving an indication of the selection of said deferral action;

code for storing state information comprising:

a position in said MUTA; and

information related to completion of said MUTA; and

code for generating a resumption object comprising an identifier of a resource for completion of said MUTA.

10. The computer program product according to claim 9 wherein said code for storing state information comprises:

code for storing state information at plural steps in said MUTA.

11. The computer program product according to claim 10 wherein said code for storing state information comprises:

code for monitoring an elapsed time since a transaction page was provided;

code for determining whether said elapsed time exceeds a predetermined time limit; and

code for storing state information if a user response is received within said predetermined time limit.

12. A computer-implemented method for completing a previously-deferred MUTA, said computer-implemented method comprising:

providing a resumption object to a client application, said resumption object comprising an identifier of a resource for completion of said MUTA;

receiving a request for said resource for completion of said MUTA

retrieving previously-received information for returning to a state at which said MUTA was previously deferred; and

providing a resource corresponding to said state at which said MUTA was previously deferred.

13. An apparatus comprising a processor, a storage, and an input-output system, said apparatus configured for completing a previously-deferred MUTA, said apparatus comprising:

a data storage system configured for storing a state object; and

server process logic configured to

provide a resumption object to a client application, said resumption object comprising an identifier of a resource for completion of said MUTA;

receive a request for said resource for completion of said MUTA

retrieve previously-received information from said state object for returning to a state at which said MUTA was previously deferred; and

provide a resource corresponding to said state at which said MUTA was previously deferred.

14. A computer program product comprising a computer readable storage medium having computer readable code embodied therein for completing a previously-deferred MUTA, said computer readable code comprising:

code for providing a resumption object to a client application, said resumption object comprising an identifier of a resource for completion of said MUTA;

code for receiving a request for said resource for completion of said MUTA

code for retrieving previously-received information for returning to a state at which said MUTA was previously deferred; and

code for providing a resource corresponding to said state at which said MUTA

was previously deferred.

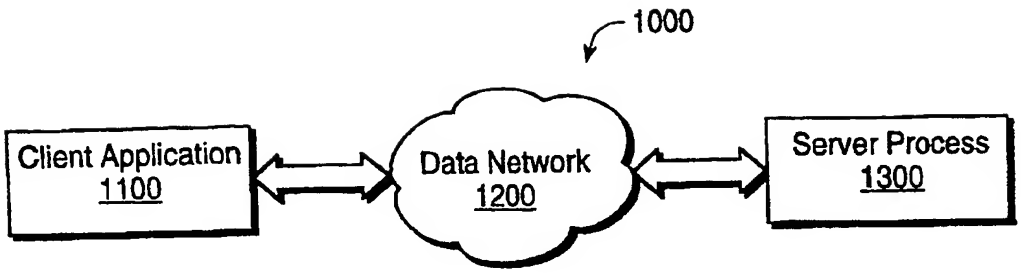
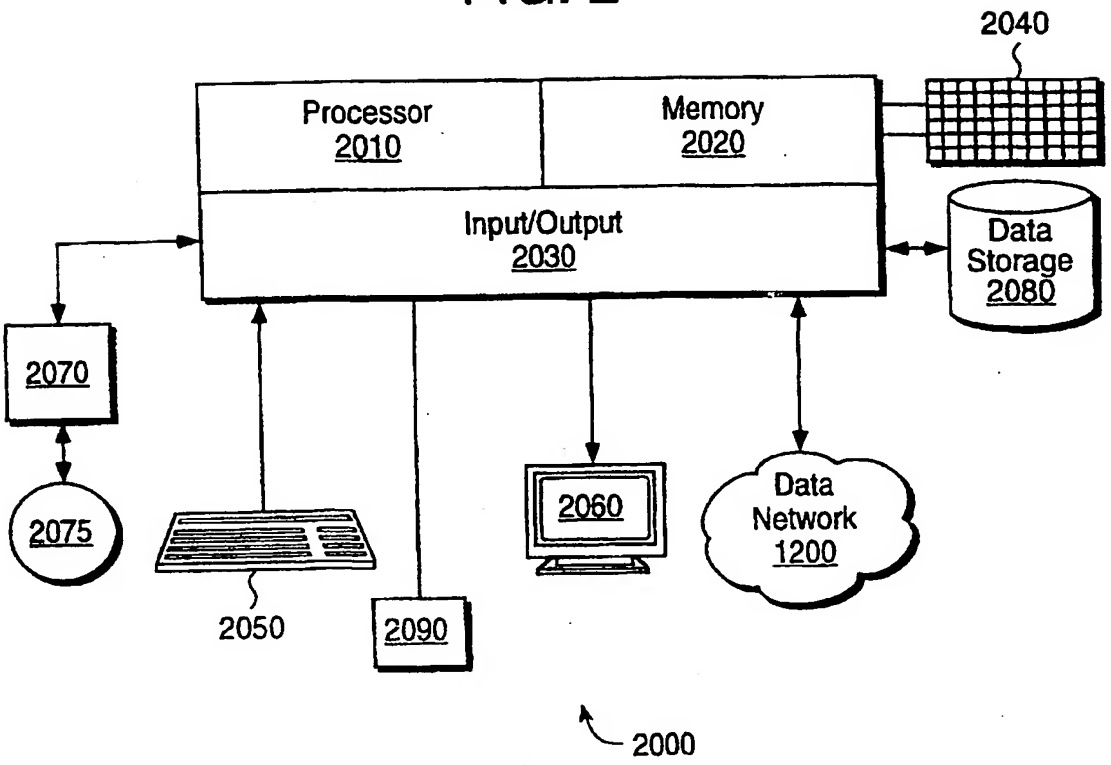

1000
Client Application
1100
Data Network
1200
Server Process
1300

FIG. 1

FIG. 2


2040
Processor
2010
Memory
2020
Input/Output
2030
Data
Storage
2080
2070
2075
2050
2090
2060
Data
Network
1200
2000

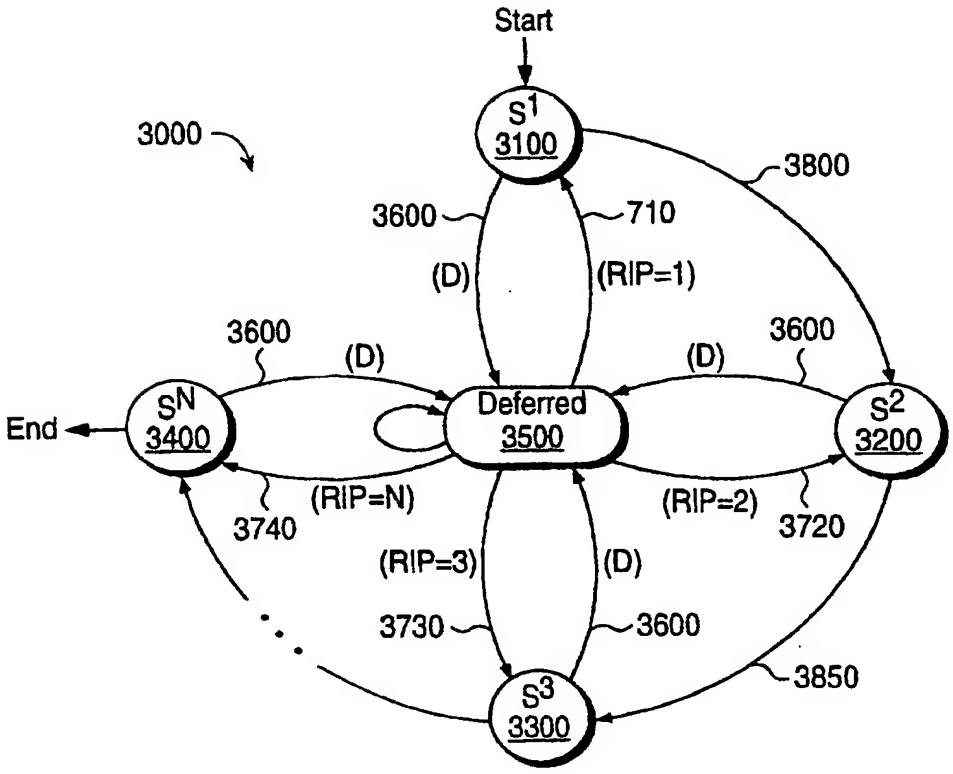

Start
3000
S1
3100
3800
3600
710
(D)
(RIP=1)
3600
(D)
(D)
3600
End
SN
3400
Deferred
3500
S2
3200
(RIP=N)
3740
(RIP=2)
3720
(RIP=3)
(D)
3730
3600
3850
S3
3300

FIG. 3

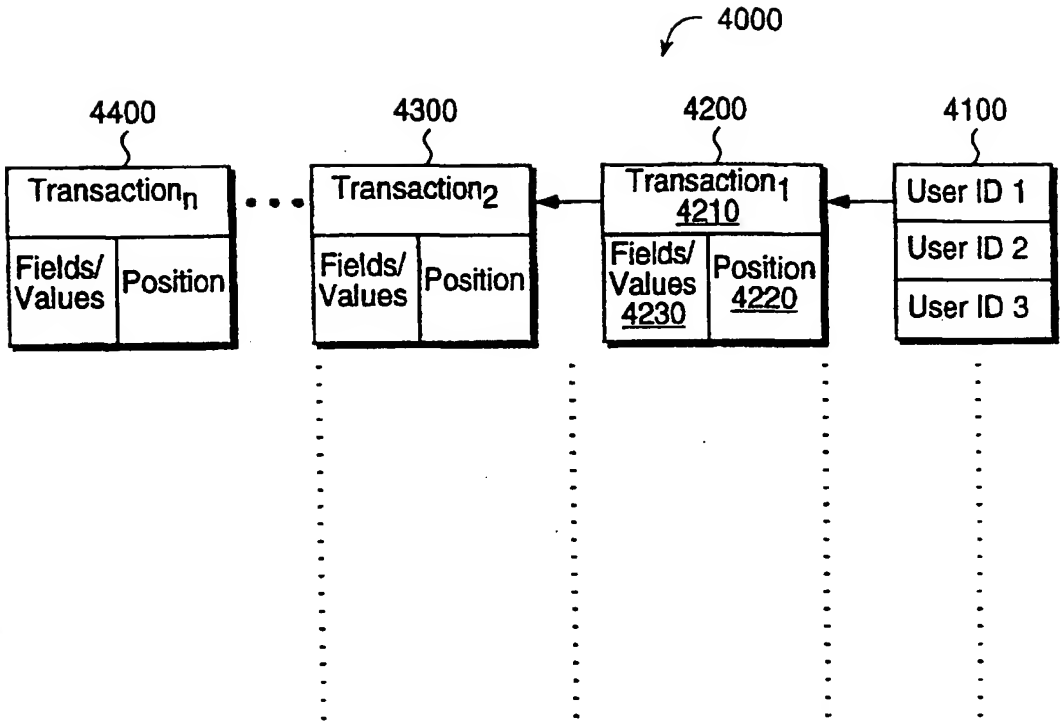

4000
4400
4300
4200
4100
Transaction$_n$
Fields/ Values
Position
Transaction$_2$
Fields/ Values
Position
Transaction$_1$ 4210
Fields/ Values 4230
Position 4220
User ID 1
User ID 2
User ID 3

FIG. 4

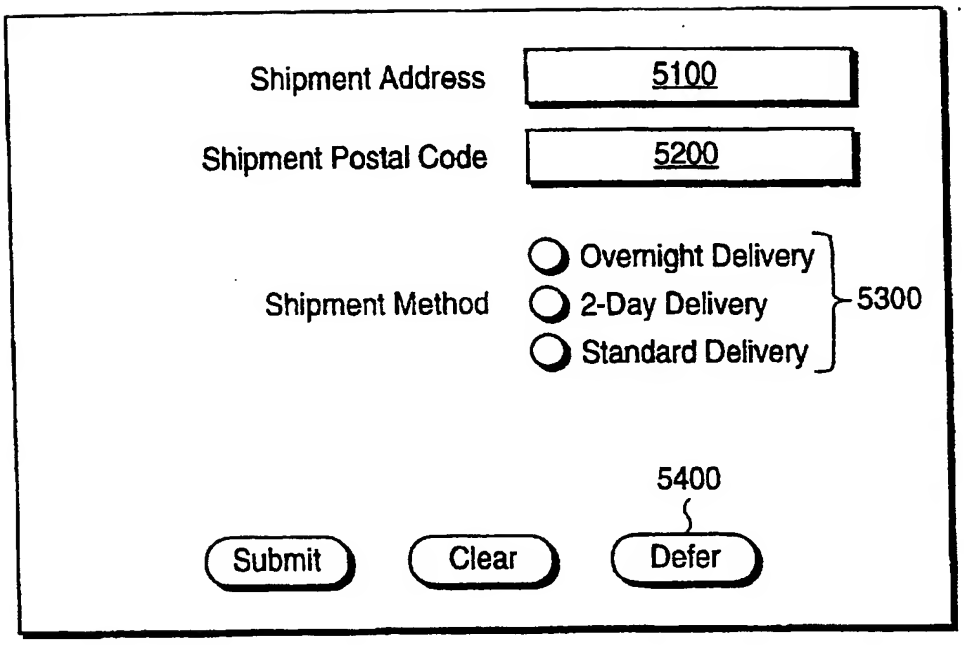

Shipment Address
5100
Shipment Postal Code
5200
Shipment Method
Overnight Delivery
2-Day Delivery
Standard Delivery
5300
5400
Submit
Clear
Defer

FIG. 5

FIG. 6

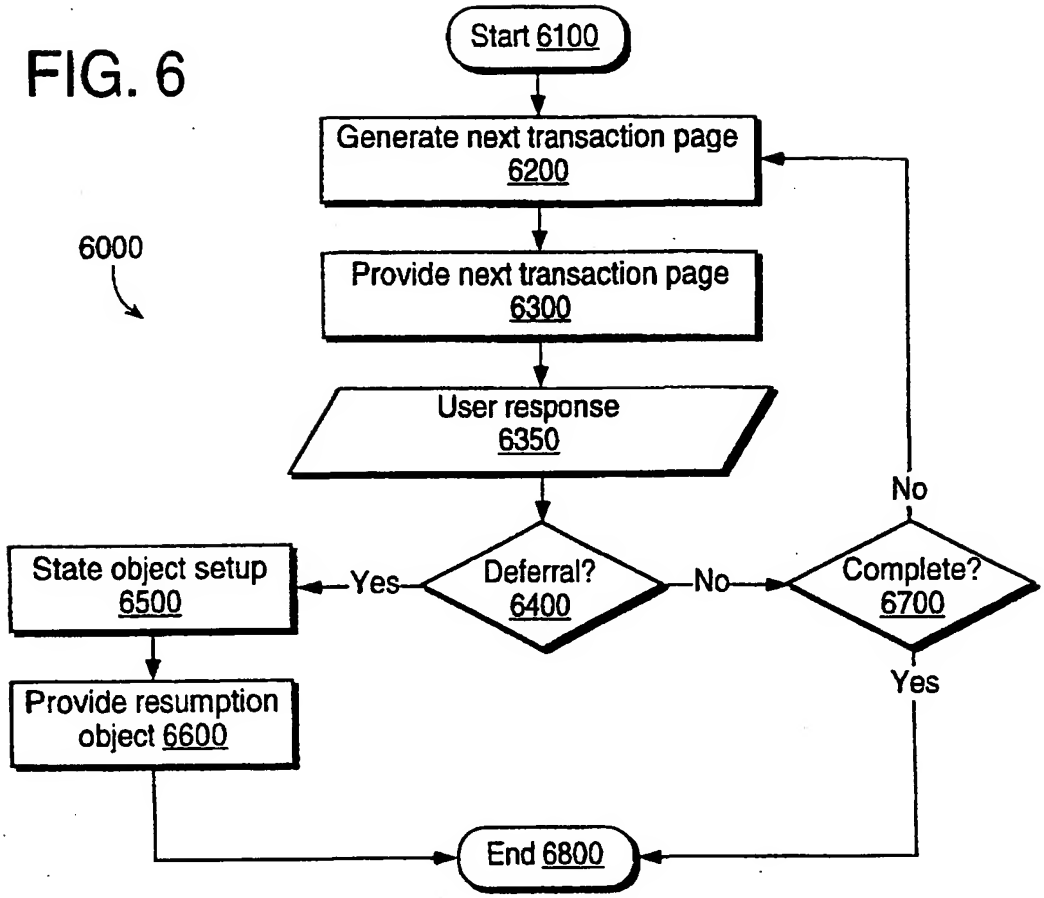
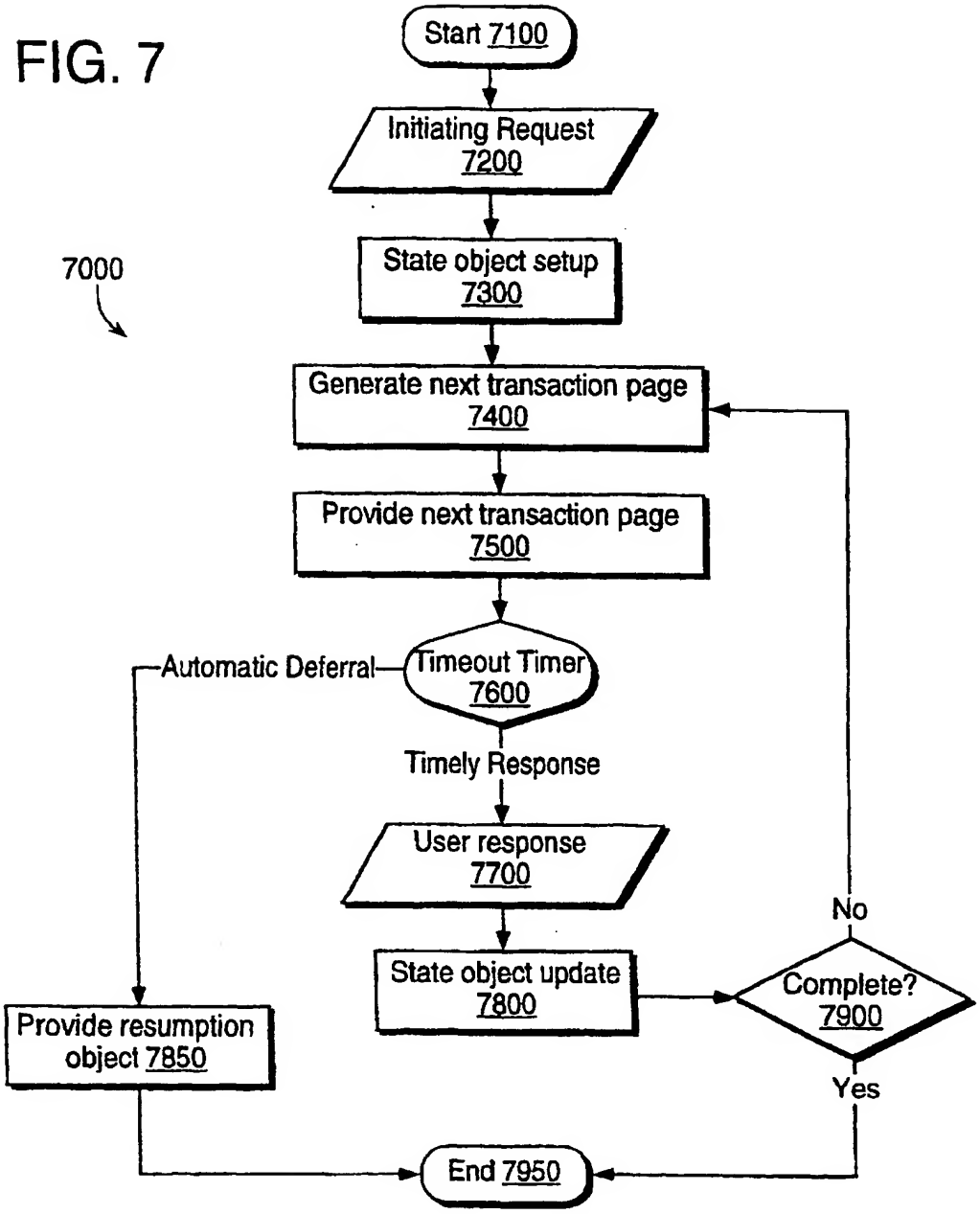

6000
Start 6100
Generate next transaction page 6200
Provide next transaction page 6300
User response 6350
Deferral? 6400
Yes
No
State object setup 6500
Provide resumption object 6600
Complete? 6700
No
Yes
End 6800

FIG. 7


7000
Start 7100
Initiating Request 7200
State object setup 7300
Generate next transaction page 7400
Provide next transaction page 7500
Timeout Timer 7600
Automatic Deferral
Timely Response
User response 7700
State object update 7800
Complete? 7900
No
Yes
Provide resumption object 7850
End 7950

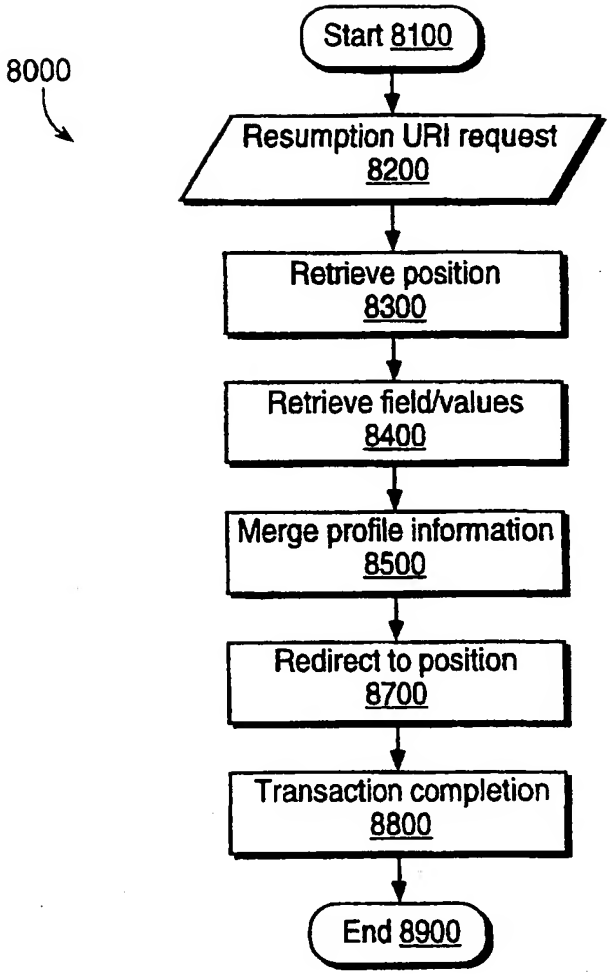

8000
Start 8100
Resumption URI request 8200
Retrieve position 8300
Retrieve field/values 8400
Merge profile information 8500
Redirect to position 8700
Transaction completion 8800
End 8900

FIG. 8

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 February 2001 (08.02.2001)

PCT

(10) International Publication Number
WO 01/09714 A3

(51) International Patent Classification[7]: G06F 9/46

(21) International Application Number: PCT/US00/19552

(22) International Filing Date: 28 July 2000 (28.07.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/364,170 29 July 1999 (29.07.1999) US
09/363,978 29 July 1999 (29.07.1999) US

(71) Applicant: RESPONDTV, INC. [US/US]; 1083 Mission Street, San Francisco, CA 94103 (US).

(72) Inventors: WEBER, Jay, C.; 302 Pope Street, Menlo Park, CA 94025 (US). LASH, Todd; 6668 Colton Boulevard, Oakland, CA 94611 (US). STEFANAC, Suzanne; 3435 Cesar Chavez, Suite PH, San Francisco, CA 94110 (US).

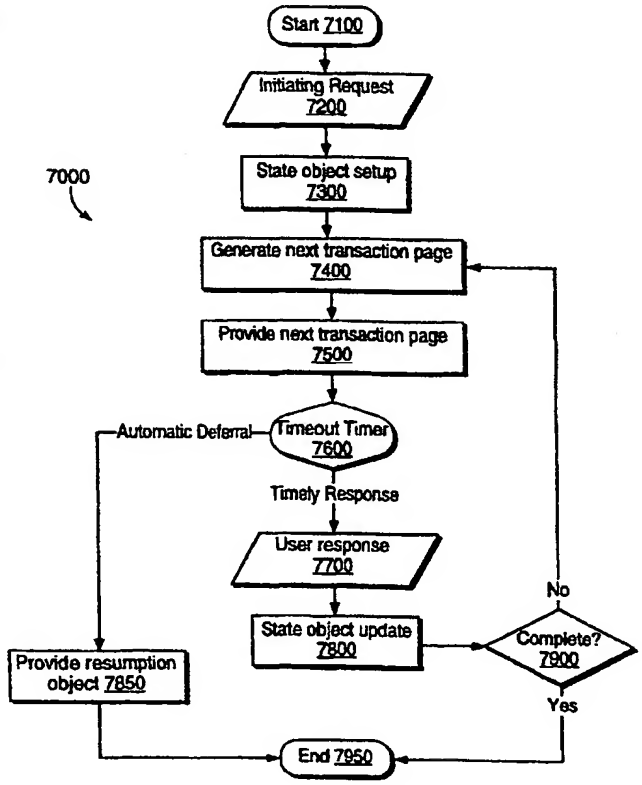
(74) Agent: DONNELLY, Darren, E.; McCutchen, Doyle, Brown & Enersen, LLP, Three Embarcadero Center, San Francisco, CA 94111 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE,

[Continued on next page]

(54) Title: METHOD, APPARATUS, AND COMPUTER PROGRAM PRODUCT FOR DEFERRED COMPLETION OF MULTI-STEP USER TRANSACTION APPLICATIONS



(57) Abstract: Disclosed are server features for allowing a client user to defer completion of multi-step user transaction applications ("MUTAs") and later resume the MUTA. One disclosed feature is providing a selectable deferral action in a page implementing part of the MUTA, e.g., HTML, XML, JavaScript/ECMA Script document. If the client user selects to defer completion, a state object is created by the server and stores state information including information previously entered during the MUTA. The server provides a resumption object to the user including a resource for resuming the deferred MUTA. When the user selects the resource, state information stored in the state object is loaded and the use may complete the remainder of the MUTA. An additionally disclosed aspect is automatic state saving where state information is stored automatically during multiple steps in the MUTA; the user then need not select deferral.

WO 01/09714 A3

IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *With international search report.*

(88) Date of publication of the international search report:
5 July 2001

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

INTERNATIONAL SEARCH REPORT

Interr. nal Application No
PCT/US 00/19552

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F9/46

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
IBM-TDB, WPI Data, EPO-Internal, INSPEC, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MAURICE SZMURLO <SZMURLO@INFO.UNICAEN.FR>, JACQUES MADELAINE <JACQUES.MADELAINE@INFO.UNICAEN.FR>: "A network of asynchronous micro-servers as a framework for server development" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 29, no. 8-13, September 1997 (1997-09), pages 1041-1051, XP002158003 Amsterdam, The Netherlands	1,2,4-6, 8-10, 12-14
A	page 1042, left-hand column, line 26 - line 46 page 1044, right-hand column, line 43 -page 1047, left-hand column, line 40 page 1048, right-hand column, last line -page 1050, left-hand column, line 4	3,7,11

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 January 2001	02/02/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Ecolivet, S.

1

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Intern. nal Application No
PCT/US 00/19552

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
X	CHRISTOPH LAMETER <CLAMETER@DEBIAN.ORG>, NICOLAS LICHTMAIER <NICK@FEEDBACK.COM.AR>, JAMES TROUP <J.J.TROUP@COMP.BRAD.AC.UK>: "lftp - Sophisticated ftp program" INTERNET DOCUMENT: LFTP MANPAGE, 'Online! 17 November 1998 (1998-11-17), XP002158004 Retrieved from the Internet: <URL:http://www.dca.fee.unicamp.br/cgi-bin /man2html/n/net/man/man1/lftp.1> 'retrieved on 2001-01-18! page 1, line 15 - line 30	1,2,5,6, 9,10
A	WO 99 23558 A (ORACLE CORP) 14 May 1999 (1999-05-14) abstract column 3, line 29 -column 4, line 10 figures 7C,7F-7I	1-14

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter nal Application No
PCT/US 00/19552

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9923558 A	14-05-1999	AU 1207299 A	24-05-1999

Form PCT/ISA/210 (patent family annex) (July 1992)

BNSDOCID: <WO_____0109714A3_I_>